ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΡΟΧΩΡΗΜΕΝΑ ΘΕΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ

Εξαμηνιαία Εργασία

Φίλιππος Σεβαστάκης, 03119183

GitHub Repository: Click here

Ιανουάριος 2024

Ζητούμενο 1

Για το πρώτο ζητούμενο της εργασίας ζητείται να εγκατασταθεί και διαμορφωθεί κατάλληλα η πλατφόρμα εκτέλεσης Apache Spark, ώστε να εκτελείται πάνω από τον διαχειριστή πόρων του Apache Hadoop, YARN, καθώς και να διαμορφωθεί κατάλληλα το Apache Hadoop Distributed File System. Για τον σκοπό αυτόν ακολούθησα τον οδηγό που δώθηκε και περιλαμβάνει την εγκατάσταση και διαμόρφωση τόσο των προαναφερθέντων, όσο και των εικονικών μηχανών του IaaS ~okeanos, που επέλεξα να χρησιμοποιήσω. Επιπλέον των βημάτων του οδηγού χρειάστηκε να συλλέξω όλα τα datasets και να τα εισάγω στο HDFS, διασφαλίζοντας παράλληλα την κατανομή τους σε όλους τους datanodes (rebalancing). Οι παραπάνω διαδικασίες περιγράφονται λεπτομερώς στο README του Setup στο συνημμένο GitHub link.

Ζητούμενο 2

Στο σημείο αυτό ζητείται να δημιουργηθεί ένα DataFrame από το βασικό σύνολο δεδομένων, διατηρώντας τα αρχικά ονόματα στηλών, αλλά προσαρμόζοντας τους τύπους των δεδομένων ως εξής:

- Date Rptd: date
- DATE OCC: date
- Vict Age: integer
- LAT: double
- LON: double

Επιπλέον, ζητείται να τυπωθεί ο συνολικός αριθμός γραμμών του συγκεκριμένου συνόλου δεδομένων, και ο τύπος κάθε στήλης.

Κώδικας

Ο κώδικας που υλοποιεί τα παραπάνω είναι ο εξής:

```
1  from pyspark.sql import SparkSession
2  from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
   ↪  DateType
3  from pyspark.sql.functions import col, to_date, unix_timestamp
4
5  spark = SparkSession \
6          .builder \
7          .appName("DataFrame") \
8          .getOrCreate()
9
10 crime_incidents_2010_to_2019_df = spark.read.format('csv') \
11         .options(header = True, inferSchema = True) \
12         .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")
13
```

```
crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")


crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
        .select(
            to_date(unix_timestamp(col("Date Rptd"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("Date Rptd"),
            to_date(unix_timestamp(col("DATE OCC"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("DATE OCC"),
            col('Vict Age').cast(IntegerType()),
            col('LAT').cast(DoubleType()),
            col('LON').cast(DoubleType()),
            )

#crime_incidents_df.show(10)
rows = crime_incidents_df.count()
print(f"DataFrame Rows count : {rows}")
crime_incidents_df.printSchema()
```

Listing 1: Query 0 - DataFrame

Έξοδος

Η έξοδος που προκύπτει από τον παραπάνω κώδικα είναι:

```
DataFrame Rows count : 2817465

|-- Date Rptd: date (nullable = true)
|-- DATE OCC: date (nullable = true)
|-- Vict Age: integer (nullable = true)
|-- LAT: double (nullable = true)
|-- LON: double (nullable = true)
```

Ζητούμενο 3

Για το ζητούμενο αυτό καλούμαστε να υλοποιήσουμε το Query 1, χρησιμοποιώντας τόσο το DataFrame όσο και το SQL API του Apache Spark, να το εκτελέσουμε με 4 Spark executors και να σχολιάσουμε αν υπάρχει διαφορά στην επίδοση μεταξύ των δύο APIs;

Query

Να βρεθούν, για κάθε έτος, οι 3 μήνες με τον υψηλότερο αριθμό καταγεγραμμένων εγκλημάτων. Ζητείται να τυπωθούν ανά έτος οι συγκεκριμένοι μήνες, ο συνολικός αριθμός εγκληματικών δράσεων που καταγράφηκαν τότε, καθώς και η θέση του συγκεκριμένου μήνα στην κατάταξη μέσα του αντίστοιχου έτους. Τα αποτελέσματα να δοθούν σε σειρά αύξουσα ως προς το έτος και φθίνουσα ως προς τον αριθμό καταγραφών.

Κώδικας

Οι κώδικες που υλοποιούν το παραπάνω, με χρήση του DataFrame και του SQL API αντίστοιχα, είναι οι εξής:

```
### Query 1 with DataFrame API ###

from pyspark.sql.window import Window
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, to_date, unix_timestamp, year, month, count,
↪  row_number
import time

start = time.time()

spark = SparkSession \
        .builder \
        .appName("Query 1 DF") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")

crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
        .select(
```

```
            to_date(unix_timestamp(col("DATE OCC"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("DATE OCC"),
            )


year_window = Window.partitionBy("year").orderBy(col("crime_total").desc())

query_1_df = crime_incidents_df \
        .withColumn("year", year("DATE OCC")) \
        .withColumn("month", month("DATE OCC")) \
        .groupBy("year", "month").agg(count("*").alias("crime_total")) \
        .sort(col("year")) \
        .withColumn("#", row_number().over(year_window)) \
        .filter(col("#") <= 3)


query_1_df.show(45)

end = time.time()
print("Query 1 DF - Execution Time: ",(end-start), "s")
```

Listing 2: Query 1 using DataFrame API

```
### Query 1 with SQL API ###

from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, to_date, unix_timestamp, year, month
import time

start = time.time()

spark = SparkSession \
        .builder \
        .appName("Query 1 SQL") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")

```

```
crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
        .select(
            to_date(unix_timestamp(col("DATE OCC"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("date_occ"),
            )


crime_incidents_df.createOrReplaceTempView("crime_incidents")

query_1 = "SELECT year, month, crime_total, rank FROM (SELECT year(date_occ) AS year,
↪  month(date_occ) AS month, count(*) AS crime_total, RANK() OVER (PARTITION BY
↪  year(date_occ) ORDER BY count(*) DESC) AS rank FROM crime_incidents GROUP BY year,
↪  month) ranked WHERE rank <=3  ORDER BY year, rank"

spark.sql(query_1).show(45)

end = time.time()
print("Query 1 SQL - Execution Time: ",(end-start), "s")
```

Listing 3: Query 1 using DataFrame API

Job submission

Για την εκτέλεση αυτών με 4 Spark executors χρειάζεται να συμπεριλάβουμε στην εντολή *spark-submit* την επιλογή *-num-executors 4*.

Output

Τα (κοινά) αποτελέσματα του Query που προκύπτουν είναι τα παρακάτω:

```
+----+-----+-----------+---+
|year|month|crime_total|  #|
+----+-----+-----------+---+
|2010|    1|      11928|  1|
|2010|    3|      11803|  2|
|2010|    4|      11793|  3|
|2011|    1|      24785|  1|
|2011|    8|      24543|  2|
|2011|    3|      24496|  3|
|2012|    1|      12969|  1|
|2012|   10|      12781|  2|
|2012|    8|      12379|  3|
|2013|    8|      11361|  1|
|2013|    7|      11009|  2|
|2013|    5|      10636|  3|
```

```
|2014|    7|      11954|  1|
|2014|   12|      11944|  2|
|2014|   10|      11933|  3|
|2015|    1|       1260|  1|
|2015|    2|        782|  2|
|2015|   12|        631|  3|
|2016|    8|      20596|  1|
|2016|    7|      20414|  2|
|2016|   10|      20037|  3|
|2017|    7|      40354|  1|
|2017|   10|      40206|  2|
|2017|    1|      39738|  3|
|2018|    8|        850|  1|
|2018|    7|        572|  2|
|2018|    1|        504|  3|
|2019|    7|      19123|  1|
|2019|    8|      18979|  2|
|2019|    3|      18858|  3|
|2020|    1|       4403|  1|
|2020|    2|       4210|  2|
|2020|    5|       4089|  3|
|2021|   10|      29461|  1|
|2021|    7|      28725|  2|
|2021|    8|      28201|  3|
|2022|    8|      14925|  1|
|2022|   12|      14289|  2|
|2022|    5|      14079|  3|
|2023|   10|      49696|  1|
|2023|    8|      49361|  2|
|2023|    7|      49358|  3|
+----+-----+-----------+---+
```

Σχολιασμός Χρόνων Εκτέλεσης[1]

Επιπλέον των αποτελεσμάτων του Query εκτυπώνουμε και τον χρόνο εκτέλεσης των δύο υλοποιήσεων. Για ακρίβεια στις μετρήσεις χρειάστηκε να τρέξουμε πολλαπλές φορές τα παραπάνω και να ληφθούν οι μέσοι όροι των χρόνων εκτέλεσης, οι οποίοι και παρατίθεται στο ακόλουθο διάγραμμα:

[1]Ως χρόνους εκτέλεσης, τόσο γι' αυτό το ζητούμενο όσο και για επόμενα, έχουν ληφθεί οι χρόνοι εκτέλεσης όλης της "εφαρμογής", συμπεριλαμβανομένων των loading times των DataFrames (ή του RDD σε επόμενο ζητούμενο)

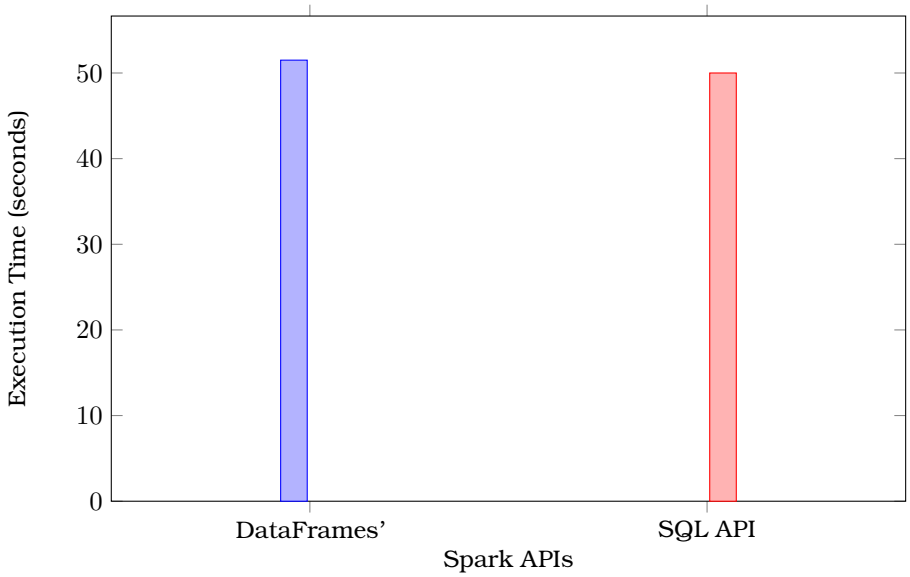

Figure 1: Query 1 Execution times: DataFrames' vs SQL APIs

Όπως παρατηρούμε οι (μέσοι) χρόνοι εκτέλεσης των δύο συμπίπτουν. Αυτό οφείλεται στο ότι τα DataFrame και SQL APIs είναι απλώς διαφορετικά interfaces, δηλαδή έχουν μεν διαφορετική σύνταξη, αλλά και τα δύο χρησιμοποιούν την ίδια execution engine (Spark Core), αναγνωρίζουν με τον ίδιο τρόπο τις δομές δεδομένων (το spark "εσωτερικά" μετατρέπει όλες τις δομές σε RDDs) και εκτελούν optimizations.

Ζητούμενο 4

Για το ερώτημα αυτό ζητείται να υλοποιήσουμε το Query 2 χρησιμοποιώντας τα DataFrame/SQL και RDD API και να αναφέρουμε και συγκρίνουμε τους χρόνους εκτέλεσης για 4 Spark executors.

Query

Να ταξινομηθούν τα τμήματα της ημέρας ανάλογα με τις καταγραφές εγκλημάτων που έλαβαν χώρα στο δρόμο (STREET), με φθίνουσα σειρά, Θεωρώντας τα εξής τμήματα μέσα στη μέρα:

- Πρωί: 5.00πμ - 11.59πμ
- Απόγευμα: 12.00μμ - 4.59μμ
- Βράδυ: 5.00μμ - 8.59μμ
- Νύχτα: 9.00μμ - 3.59πμ

Κώδικας

Οι κώδικες για τις ζητούμενες υλοποιήσεις είναι οι εξής:

```
### Query 2 with DataFrame API ###

from pyspark.sql.window import Window
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, when
import time

start = time.time()

spark = SparkSession \
        .builder \
        .appName("Query 2 DF") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")


crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
```

```
        .select(
            col("TIME OCC").cast("int"),
            col("Premis Desc"),
            )

street_crime_incidents_df = crime_incidents_df.filter(col("Premis Desc") == "STREET")


query_2_df = street_crime_incidents_df \
        .withColumn(
                "time of day",
                when((col("TIME OCC") >= 500) & (col("TIME OCC") < 1200), "Morning") \
                .when((col("TIME OCC") >= 1200) & (col("TIME OCC") < 1700), "Afternoon") \
                .when((col("TIME OCC") >= 1700) & (col("TIME OCC") < 2100), "Evening") \
                .otherwise("Night")
                ) \
        .groupBy("time of day").count() \
        .orderBy(col("count").desc())

query_2_df.show()

end = time.time()
print("Query 2 DF - Execution Time: ",(end-start), "s")
```

Listing 4: Query 2 using DataFrame API

```
### Query 2 with RDD API ###

import re
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType, DateType
from pyspark.sql.functions import col, when
import time

start = time.time()

spark = SparkSession \
        .builder \
        .appName("Query 2 RDD") \
        .getOrCreate() \
        .sparkContext

crime_incidents_rdd = spark.textFile("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv,hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")
```

```
18
19
20  def time_of_day(hour):
21      if 500 <= hour < 1200:
22          return "Morning"
23      elif 1200 <= hour < 1700:
24          return "Afternoon"
25      elif 1700 <= hour < 2100:
26          return "Evening"
27      else:
28          return "Night"
29
30  def comma_splits(x):
31      return re.split(r',(?=(?:[^\"]*\"[^\"]*\")*[^\"]*$)', x)
32
33  query_2_rdd = crime_incidents_rdd.map(comma_splits) \
34          .filter(lambda x: x[15] == "STREET") \
35          .map(lambda x: (time_of_day(int(x[3])), 1)) \
36          .reduceByKey(lambda x,y: x+y) \
37          .sortBy(lambda x: x[1], ascending=False)
38
39  print(query_2_rdd.collect())
40
41  end = time.time()
42  print("Query 2 RDD - Execution Time: ",(end-start), "s")
```

Listing 5: Query 2 using RDD's API

Job submission

Όπως και στο προηγούμενο ζητούμενο ζητείται εκτέλεση των παραπάνω υλοποιήσεων με 4 Spark executors, οπότε εκτελούμε τα scripts με τις εξής εντολές:
spark-submit –num-executors 4 query_2_df.py
spark-submit -num-executors 4 query_2_rdd.py

Output

Οι έξοδοι που παράγονται είναι, αντίστοιχα, οι εξής:

```
+-----------+------+
|time of day| count|
+-----------+------+
|      Night|227943|
|    Evening|179359|
|  Afternoon|142466|
|    Morning|119213|
+-----------+------+

Query 2 DF - Execution Time:  72.95180583000183 s
```

Listing 1: DataFrame's Output

```
[('Night', 227943), ('Evening', 179359), ('Afternoon', 142466), ('Morning', 119213)]
Query 2 RDD - Execution Time:  74.37339091300964 s
```

Listing 2: RDD's Output

Σχολιασμός Χρόνων Εκτέλεσης

Εκτελούμε τα παραπάνω πολλαπλές φορές και παίρνουμε, όπως πριν, έναν μέσο όρο των χρόνων εκτέλεσης της κάθε υλοποίησης, οι οποίοι και παρατίθενται παρακάτω:

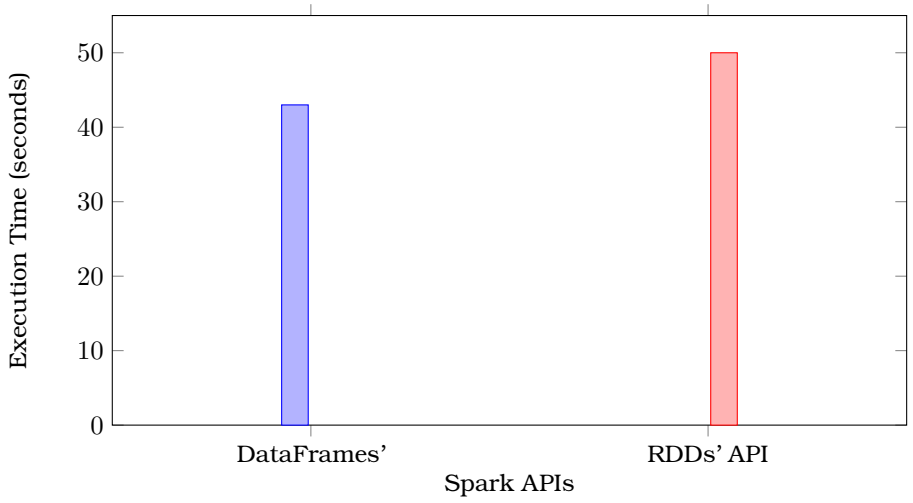

Figure 2: Query 2 Execution times: DataFrames' vs RDDs' APIs

Όπως μπορούμε να διακρίνουμε από το γράφημα, οι δύο αυτές υλοποίσεις διαφέρουν ως προς τους χρόνους εκτέλεσής τους. Τα RDDs είναι η "primitive" δομή δεδομένων της Spark (όλα τα δεδομένα

ανεξαρτήτως API "μετατρέπονται" σε RDDs, δηλαδή εσωτερικά το spark εκτελεί transformations μόνο μεταξύ RDDs). Στην περίπτωση του RDDs' API επεξεργαζόμαστε απευθείας τα δεδομένα ως RDDs - με τις διάφορες μεθόδους .filter, .map, etc - (low-level API) και κατα συνέπεια δεν γίνονται optimizations. Αντιθέτως, το DataFrames' API, όπως και το SQL API, (high-level APIs) χρησιμοποιούν optimizations (π.χ. Adaptive Query Execution), οδηγώντας σε καλύτερο performance, όπως συμπεραίνεται και παραπάνω.

Ζητούμενο 5

Για το ζητούμενο αυτό καλούμαστε να υλοποιήσουμε το Query 3 με DataFrame/SQL APIs και να το εκτελέσουμε με 2, 3 και 4 Spark executors, συγκρίνοντας κατόπιν τους χρόνους εκτέλεσης αυτών.

Query

Να βρεθεί η καταγωγή (descent) των καταγεγραμμένων θυμάτων εγκλημάτων στο Los Angeles για το έτος 2015 στις 3 περιοχές (ZIP Codes) με το υψηλότερο και τις 3 περιοχές (ZIP Codes) με το χαμηλότερο εισόδημα ανά νοικοκυριό. Τα αποτελέσματα να τυπωθούν από το υψηλότερο στο χαμηλότερο αριθμό θυμάτων ανά φυλετικό γκρουπ.

Κώδικας

Ο κώδικας που υλοποιεί το παραπάνω είναι ο εξής:

```
### Query 3 ###

from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, split, regexp_replace, desc, year, to_date,
↪  unix_timestamp, when
import time

start = time.time()

spark = SparkSession \
        .builder \
        .appName("Query 3") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")
```

```
22
23  crime_incidents_df = crime_incidents_2010_to_2019_df \
24          .union(crime_incidents_2020_to_curr_df) \
25
26  LA_income_2015_df = spark.read.format('csv') \
27          .options(header = True, inferSchema = True) \
28          .load("hdfs://okeanos-master:54310/data/LA_income_2015.csv")
29
30  rev_geocoding_df = spark.read.format('csv') \
31          .options(header = True, inferSchema = True) \
32          .load("hdfs://okeanos-master:54310/data/revgecoding.csv")
33
34  rev_geocoding_unique_df = rev_geocoding_df.withColumn("ZIPcode", split(col("ZIPcode"),
    ↪  ";").getItem(0))\
35
36  crime_incidents_2015_df = crime_incidents_df.select(col("Vict Descent"), col("LAT"),
    ↪  col("LON"), col("DATE OCC")) \
37          .withColumn("year", year(to_date(unix_timestamp(col("DATE OCC"), "MM/dd/yyyy
            ↪  hh:mm:ss a").cast("timestamp"),"yyyy-MM-dd"))) \
38          .filter((col("year") == 2015) & (col("Vict Descent").isNotNull()))
39
40
41  crime_incidents_2015_zip_df = crime_incidents_2015_df.join(rev_geocoding_unique_df,
    ↪  ['LAT', 'LON']) \
42          .select(col("Vict Descent"), col("ZIPcode"))
43  # maybe use .persist()
44
45  distinct_zip_codes_df = crime_incidents_2015_zip_df.select("ZIPcode").distinct()
46
47
48  LA_income_formatted_df = LA_income_2015_df.select(col("Zip Code").alias("ZIPcode"),
    ↪  col("Estimated Median Income")) \
49          .withColumn("Estimated Median Income",regexp_replace("Estimated Median Income",
            ↪  "[$,]", "").cast("integer"))
50
51  distinct_LA_incomes_df = LA_income_formatted_df.join(distinct_zip_codes_df, "ZIPcode") \
52  # maybe use .persist()
53
54  best_worst_3_df = distinct_LA_incomes_df.orderBy(col("Estimated Median
    ↪  Income")).limit(3) \
55          .union(distinct_LA_incomes_df.orderBy(col("Estimated Median
            ↪  Income").desc()).limit(3))
56
57
58  query_3_df = crime_incidents_2015_zip_df.join(best_worst_3_df, "ZIPcode") \
59          .withColumn(
60                      "Victim Descent",
61                      when(col("Vict Descent") == "A", "Other Asian") \
```

```
                    .when(col("Vict Descent") == "B", "Black") \
                    .when(col("Vict Descent") == "C", "Chinese") \
                    .when(col("Vict Descent") == "D", "Cambodian") \
                    .when(col("Vict Descent") == "F", "Filipino") \
                    .when(col("Vict Descent") == "G", "Guamanian") \
                    .when(col("Vict Descent") == "H", "Hispanic/Latin/Mexican") \
                    .when(col("Vict Descent") == "I", "American Indian/Alaskan Native") \
                    .when(col("Vict Descent") == "J", "Japanese") \
                    .when(col("Vict Descent") == "K", "Korean") \
                    .when(col("Vict Descent") == "L", "Laotian") \
                    .when(col("Vict Descent") == "P", "Pacific Islander") \
                    .when(col("Vict Descent") == "S", "Samoan") \
                    .when(col("Vict Descent") == "U", "Hawaiian") \
                    .when(col("Vict Descent") == "V", "Vietnamese") \
                    .when(col("Vict Descent") == "W", "White") \
                    .when(col("Vict Descent") == "X", "Unknown") \
                    .when(col("Vict Descent") == "Z", "Asian Indian") \
                    .otherwise("Unknown")
                    ) \
        .groupBy(col("ZIPcode")).count() \
        .orderBy(col("count").desc()) \
        .select(col("ZIPcode"), col("count").alias("#"))

query_3_df.show()

end = time.time()
print("Query 3 - Execution Time: ",(end-start), "s")
```

Listing 6: Query 3

Output

Η έξοδος που παράγεται από τον παραπάνω κώδικα είναι:

```
+--------------------+---+
|      Victim Descent|  #|
+--------------------+---+
|               Black|223|
|               White|193|
|Hispanic/Latin/Me...|149|
|             Unknown| 68|
|         Other Asian| 17|
+--------------------+---+

Query 3 - Execution Time:  108.99271941184998 s
```

Σχολιασμός Χρόνων Εκτέλεσης

Εκτελώντας πολλαπλές φορές τον παραπάνω κώδικα με 2, 3 και 4 spark executors και υπολογίζοντας τον μέσο όρο των χρόνων εκτέλεσης για την κάθε περίπτωση καταλήγουμε στα παρακάτω αποτελέσματα:

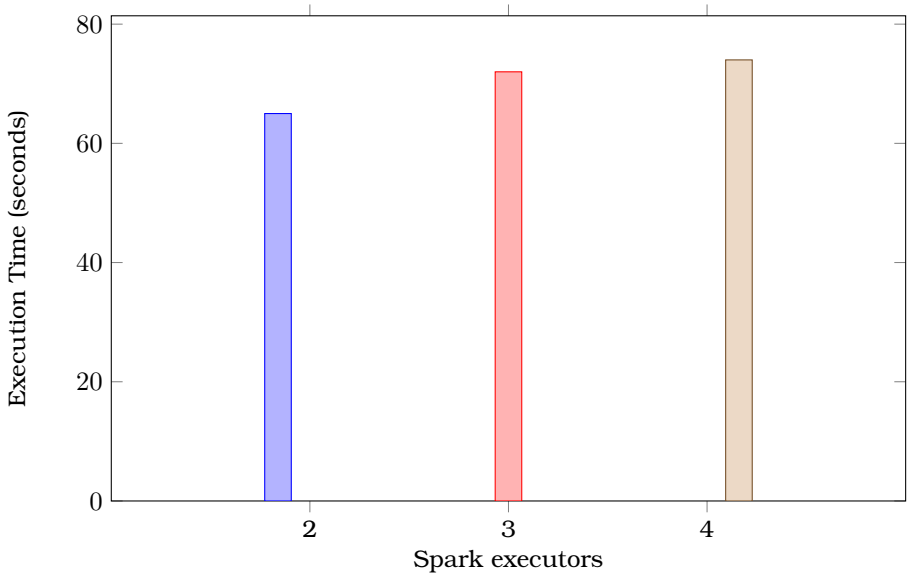

Figure 3: Query 3 Execution times for 2, 3 and 4 Spark executors

Στο παραπάνω διάγραμμα παρατηρούμε ότι ο χρόνος εκτέλεσης του Query 3 αυξάνεται με την αύξηση των executors. Αυτή η συμπεριφορά, ωστόσο, είναι αναμενόμενη για την συγκεκριμένη υλοποίηση. Η χρήση πολλαπλών executors ανά datanode μπορεί να οδηγήσει σε καλύτερο χρόνο εκτέλεσης μόνο αν ο κώδικας (query execution) είναι τόσο παραλληλοποιήσιμος και με τόσο λίγη ανάγκη για επικοινωνία και συγχρονισμό μεταξύ των datanodes, ώστε ο χρόνος που "γλυτώνουμε" από την παραλληλοποίηση να είναι μεγαλύτερος από τον χρόνο που χάνουμε από το overhead των executors (αρχικοποίησής τους και κατανομής τους στα datanodes). Η παραπάνω υλοποίηση, όμως, είναι εύκολο να δούμε ότι δεν ανήκει σε αυτή την περίπτωση. Για τον υπολογισμό των τριων[2] joins απαιτείται επικοινωνία τόσο μεταξύ των datanodes (μη αποφευκτέο για όλες τις παραπάνω εκτελέσεις), όσο και μεταξύ των executors, αν υπάρχουν περισσότεροι του ενός σε κάποιο datanode.

[2]Χρειάστηκαν 3 joins αντί για 2 για να υπολογιστούν οι 3 περιοχές με το υψηλότερο και οι 3 με το χαμηλότερο εισόδημα *στις οποίες να υπάρχουν καταγεγραμμένα εγκλήματα*, με επιπλέον συνέπεια να μην έχουμε μέχρι και εκείνο το join κάποιο "filtering" ως προς τις περιοχές

Ζητούμενο 6

Στο σημείο αυτό ζητείται να υλοποιήσουμε το Query 4 χρησιμοποιώντας το DataFrame/SQL API.

Query

Στόχος είναι να εξεταστεί κατά πόσον τα εγκλήματα που καταγράφονται στην πόλη του Los Angeles αντιμετωπίζονται από το πλησιέστερο στον τόπο εγκλήματος αστυνομικό τμήμα ή όχι. Για το λόγο αυτό, θα εκτελέσουμε δύο ζεύγη παρόμοιων ερωτημάτων και θα συγκρίνουμε τα αποτελέσματα:

1. Liable Police Station Subquery:
   (a) Να υπολογιστεί ανά έτος ο αριθμός εγκλημάτων με καταγραφή χρήσης οποιασδήποτε μορφής πυροβόλων όπλων και η μέση απόσταση (σε km) των σημείων όπου αυτά έλαβαν χώρα **από το αστυνομικό τμήμα που ανέλαβε την έρευνα για το περιστατικό** και
   (b) ανά αστυνομικό τμήμα ο αριθμός εγκλημάτων με καταγραφή χρήσης οποιαδήποτε μορφής όπλων που του ανατέθηκε, καθώς και η μέση απόσταση του εκάστοτε τόπου εγκλήματος από το αστυνομικό τμήμα.
2. Closest Police Station Subquery:
   (a) Να υπολογιστεί ανά έτος ο αριθμός εγκλημάτων με καταγραφή χρήσης οποιασδήποτε μορφής πυροβόλων όπλων και η μέση απόσταση (σε km) των σημείων όπου αυτά έλαβαν χώρα **από το πλησιέστερο σε αυτά αστυνομικό τμήμα** και
   (b) ανά αστυνομικό τμήμα ο αριθμός εγκλημάτων με καταγραφή χρήσης οποιαδήποτε μορφής όπλων που του ανατέθηκε, καθώς και η μέση απόσταση του εκάστοτε τόπου εγκλήματος από το αστυνομικό τμήμα.

Κώδικας

Ο κώδικας για τα subqueries (a) και (b) που αφορούν στο αστυνομικό τμήμα που ανέλαβε την έρευνα (Liable Police Station) για το περιστατικό είναι ο εξής:

```
### Query 4: Liable Police Stations ###

from pyspark.sql.window import Window
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, regexp_replace, desc, year, to_date,
↪  unix_timestamp, when, udf, count, avg, format_number
from geopy.distance import geodesic

def get_distance(lat1, long1, lat2, long2):
    return geodesic((lat1, long1), (lat2, long2)).km

get_distance_udf = udf(get_distance, DoubleType())

```

```
spark = SparkSession \
        .builder \
        .appName("Query 4: Liable Police Stations") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")

LAPD_Police_Stations_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/LAPD_Police_Stations.csv") \
        .select(
            col("PREC").cast(IntegerType()),
            col("X").cast(DoubleType()),
            col("Y").cast(DoubleType()),
            col("DIVISION").alias("division")
            )

crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
        .filter((col("Weapon Used Cd").startswith("1")) & (col("LAT") != "0")) \
        .select(
            to_date(unix_timestamp(col("DATE OCC"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("DATE OCC"),
            col("LAT").cast(DoubleType()),
            col("LON").cast(DoubleType()),
            col("AREA ").cast(IntegerType()).alias("PREC")
            )

crime_x_station_df = crime_incidents_df.join(LAPD_Police_Stations_df, "PREC") \
        .withColumn("year", year("DATE OCC")) \
        .withColumn("distance", get_distance_udf(col("LAT"), col("LON"), col("Y"),
        ↪  col("X"))) \
        .persist()

query_4a_df = crime_x_station_df \
        .groupBy("year").agg(
                format_number(avg(col("distance")), 3).alias("average_distance"),
                count("*").alias("#")
                ) \
        .orderBy(col("year")) \
        .show()

```

```
query_4b_df = crime_x_station_df \
        .groupBy("division").agg(
                format_number(avg(col("distance")),3).alias("average_distance"),
                count("*").alias("#")
                ) \
        .orderBy(col("#").desc()) \
        .show(25)
```

Listing 7: Query 4: Liable Police Station

Ο κώδικας για τα subqueries (a) και (b) που αφορούν στο πλησιέστερο στο περιστατικό αστυνομικό τμήμα είναι ο εξής:

```
### Query 4: Closest Police Stations ###

from pyspark.sql.window import Window
from pyspark.sql import SparkSession
from pyspark.sql.types import IntegerType, DoubleType, StringType, TimestampNTZType,
↪  DateType
from pyspark.sql.functions import col, regexp_replace, desc, year, to_date,
↪  unix_timestamp, when, udf, count, avg, format_number, row_number,
↪  monotonically_increasing_id
from geopy.distance import geodesic

def get_distance(lat1, long1, lat2, long2):
    return geodesic((lat1, long1), (lat2, long2)).km

get_distance_udf = udf(get_distance, DoubleType())

spark = SparkSession \
        .builder \
        .appName("Query 4: Closest Police Stations") \
        .getOrCreate()

crime_incidents_2010_to_2019_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2010-2019.csv")

crime_incidents_2020_to_curr_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/crime_incidents_2020-.csv")

LAPD_Police_Stations_df = spark.read.format('csv') \
        .options(header = True, inferSchema = True) \
        .load("hdfs://okeanos-master:54310/data/LAPD_Police_Stations.csv") \
        .select(
            col("PREC").cast(IntegerType()),
```

```
            col("X").cast(DoubleType()),
            col("Y").cast(DoubleType()),
            col("DIVISION").alias("division")
            )

crime_incidents_df = crime_incidents_2010_to_2019_df \
        .union(crime_incidents_2020_to_curr_df) \
        .filter((col("Weapon Used Cd").startswith("1")) & (col("LAT") != "0")) \
        .select(
            to_date(unix_timestamp(col("DATE OCC"),"MM/dd/yyyy hh:mm:ss
            ↪  a").cast("timestamp"),"yyyy-MM-dd").alias("DATE OCC"),
            col("LAT").cast(DoubleType()),
            col("LON").cast(DoubleType()),
            col("AREA ").cast(IntegerType()).alias("PREC")
            ) \
        .withColumn("ID", monotonically_increasing_id())

crime_x_stations_df = crime_incidents_df.join(LAPD_Police_Stations_df) \
        .withColumn("year", year("DATE OCC")) \
        .withColumn("distance", get_distance_udf(col("LAT"), col("LON"), col("Y"),
        ↪  col("X")))

closest_window = Window.partitionBy("ID").orderBy("distance")

crime_x_station_df = crime_x_stations_df \
        .withColumn("number_in_partition", row_number().over(closest_window)) \
        .filter(col("number_in_partition") == 1) \
        .persist()

query_4a_df = crime_x_station_df \
        .groupBy("year").agg(
                format_number(avg(col("distance")), 3).alias("average_distance"),
                count("*").alias("#")
                ) \
        .orderBy(col("year")) \
        .show()

query_4b_df = crime_x_station_df \
        .groupBy("division").agg(
                format_number(avg(col("distance")),3).alias("average_distance"),
                count("*").alias("#")
                ) \
        .orderBy(col("#").desc()) \
        .show(25)
```

Listing 8: Query 4: Closest Police Station

Η έξοδος του πρώτου κώδικα που αφορά στα αστυνομικά τμήματα που ανέλαβαν την εκάστοτε υπόθεση είναι η εξής:

```
+----+----------------+-----+
|year|average_distance|    #|
+----+----------------+-----+
|2010|           2.681| 4594|
|2011|           2.831|10075|
|2012|           2.821| 4312|
|2013|           2.655| 4070|
|2014|           2.836| 4115|
|2015|           2.730|  104|
|2016|           2.668| 7982|
|2017|           2.748|16014|
|2018|           2.057|   45|
|2019|           2.739| 7129|
|2020|           2.478| 1989|
|2021|           2.650|14608|
|2022|           2.408| 7506|
|2023|           2.634|23309|
+----+----------------+-----+

+----------------+----------------+-----+
|        division|average_distance|    #|
+----------------+----------------+-----+
|     77TH STREET|           2.652|15681|
|       SOUTHEAST|           2.089|11434|
|          NEWTON|           2.031| 8877|
|       SOUTHWEST|           2.716| 8807|
|      HOLLENBECK|           2.656| 6383|
|          HARBOR|           4.053| 5897|
|         RAMPART|           1.578| 5215|
|         OLYMPIC|           1.811| 4164|
|         CENTRAL|           1.123| 4011|
|       NORTHEAST|           3.902| 3989|
|       HOLLYWOOD|           1.448| 3937|
|         MISSION|           4.731| 3789|
|        WILSHIRE|           2.294| 3630|
| NORTH HOLLYWOOD|           2.662| 3070|
|     WEST VALLEY|           3.415| 3011|
|        FOOTHILL|           3.823| 2924|
|        VAN NUYS|           2.223| 2725|
|         PACIFIC|           3.786| 2368|
|      DEVONSHIRE|           3.984| 2240|
|         TOPANGA|           3.475| 2068|
|WEST LOS ANGELES|           4.275| 1632|
+----------------+----------------+-----+
```

Η έξοδος του δεύτερου κώδικα που αφορά στο δεύτερο subquery, για τα πλησιέστερα αστυνομικά τμήματα, είναι η εξής:

```
+----+----------------+-----+
|year|average_distance|    #|
+----+----------------+-----+
|2010|           2.353| 4594|
|2011|           2.481|10075|
|2012|           2.554| 4312|
|2013|           2.317| 4070|
|2014|           2.438| 4115|
|2015|           2.640|  104|
|2016|           2.440| 7982|
|2017|           2.387|16014|
|2018|           1.915|   45|
|2019|           2.429| 7129|
|2020|           2.231| 1989|
|2021|           2.347|14608|
|2022|           2.196| 7506|
|2023|           2.333|23309|
+----+----------------+-----+

+----------------+----------------+-----+
|        division|average_distance|    #|
+----------------+----------------+-----+
|     77TH STREET|           1.715|12848|
|       SOUTHWEST|           2.254|10668|
|       SOUTHEAST|           2.204|10521|
|        WILSHIRE|           2.442| 6547|
|          NEWTON|           1.590| 6522|
|      HOLLENBECK|           2.635| 6435|
|       HOLLYWOOD|           1.978| 5806|
|          HARBOR|           3.849| 5752|
|         OLYMPIC|           1.662| 5330|
|         RAMPART|           1.409| 5021|
|        VAN NUYS|           2.931| 4512|
|         CENTRAL|           1.007| 4091|
|        FOOTHILL|           3.595| 3605|
|       NORTHEAST|           3.739| 3160|
| NORTH HOLLYWOOD|           2.729| 3134|
|     WEST VALLEY|           2.712| 2911|
|         MISSION|           3.825| 2361|
|         PACIFIC|           3.751| 2254|
|         TOPANGA|           3.047| 2113|
|      DEVONSHIRE|           3.076| 1183|
|WEST LOS ANGELES|           2.750| 1078|
+----------------+----------------+-----+
```

Ζητούμενο 7:

Για το τελευταίο ζητούμενο καλούμαστε να χρησιμοποιήσουμε τις μεθόδους *hint* και *explain* των DataFrame/SQL APIs ώστε να εκτελεστούν με διαφορετικό τρόπο (BROADCAST, MERGE, SHUFFLE_HASH, SHUFFLE_REPLICATE_NL) και να πάρουμε το πλάνο μέσω κειμένου και γραφικά απο το Spark History UI για τα joins των Queries 3 και 4.

Join Strategies

Όπως είναι φυσικό, λόγω του διαμοιρασμού των δεδομένων στους διάφορους κόμβους, σε operations όπως τα joins χρειάζεται να υπάρξει μεταφορά δεδομένων μεταξύ των κόμβων, γιατί εγγραφές που βρίσκονται σε διαφορετικούς μπορεί να πληρούν το "condition" του join και πρέπει να (συν)υπολογιστούν. Έτσι, τα σχετικά με το join δεδομένα (έστω αυτά ανά condition) πρέπει να βρεθούν στον ίδιο κόμβο. Για τον σκοπό αυτόν υπάρχουν περισσότεροι του ενός τρόποι ("στρατηγικές"), καθένας αποδοτικός υπό συνθήκες και τον οποίο επιλέγει το optimization engine σύμφωνα με τις συνθήκες τις εφαρμογής. Αυτές τις στρατηγικές μελετάμε και συγκρίνουμε παρακάτω χρησιμοποιώντας τις μεθόδους hint (για την εκτέλεση συγκεκριμένου τρόπου join) και explain για την εμφάνιση του πλάνου εκτέλεσης του Spark.

Broadcast Join

Αν το ένα εκ των δύο dataframes του join είναι αρκετά μικρό (spark.sql.autoBroadcastJoinThreshold) ώστε να χωράει στη μνήμη των executors, τότε αρκεί να γίνει broadcasted (να μεταδοθεί σε όλους τους executors) και θα μπορέσει να πραγματοποιηθεί το join χωρίς την μεταφορά δεδομένων από το άλλο datafrane. Αυτή η "στρατηγική" λέγεται broadcast join, μπορεί να είναι πολύ αποδοτική για joins μεταξύ αρκετά μικρού και μεγάλου dataframe, ενώ "χωρίζεται" στους εξής 2 επιπλέον τρόπους εκτέλεσης:

- BroadcastHashJoin (χρησιμοποιείται όταν το condition αφορά ισότητα), κατά τον οποίο δημιουργείται ένα hash table από το μικρό dataset, το οποίο σειριοποιείται και γίνεται broadcasted σε όλους executors, γίνονται hashed και οι τιμές-κλειδιά (που αφορούν το condition) του μεγάλου dataset σε κάθε executor και εκτελείται το join σε αυτούς με records-αποτελέσματα αυτά που προκύπτουν από την σύγκριση των hashes.
- BroadcastNestedLoopJoin (χρησιμοποιείται όταν το condition αφορά ανισότητα), κατά τον οποίο το μικρό dataset σειριοποιείται και γίνεται broadcasted σε όλους τους executors, στους οποίους γίνεται επαναληπτικά ο έλεγχος του condition μεταξύ των αντίστοιχων τιμών-κλειδιών του μικρού και του μεγάλου dataset

Παρακάτω παρατίθενται τα physical plans των εκτελέσεων των Query 3 και 4 (liable και closest subquery) με μόνο broadcast hint ως μεθόδου για τα joins, αντίστοιχα:

```
== Physical Plan ==
AdaptiveSparkPlan (70)
+- Project (69)
   +- Sort (68)
      +- Exchange (67)
         +- HashAggregate (66)
            +- Exchange (65)
```

```
+- HashAggregate (64)
   +- Project (63)
      +- BroadcastHashJoin Inner BuildRight (62)
         :- Project (13)
         :  +- BroadcastHashJoin Inner BuildRight (12)
         :     :- Union (7)
         :     :  :- Project (3)
         :     :  :  +- Filter (2)
         :     :  :     +- Scan csv  (1)
         :     :  +- Project (6)
         :     :     +- Filter (5)
         :     :        +- Scan csv  (4)
         :     +- BroadcastExchange (11)
         :        +- Project (10)
         :           +- Filter (9)
         :              +- Scan csv  (8)
         +- BroadcastExchange (61)
            +- Union (60)
               :- TakeOrderedAndProject (36)
               :  +- Project (35)
               :     +- BroadcastHashJoin Inner BuildRight (34)
               :        :- Project (16)
               :        :  +- Filter (15)
               :        :     +- Scan csv  (14)
               :        +- BroadcastExchange (33)
               :           +- HashAggregate (32)
               :              +- Exchange (31)
               :                 +- HashAggregate (30)
               :                    +- Project (29)
               :                       +- BroadcastHashJoin Inner BuildRight (28)
               :                          :- Union (23)
               :                          :  :- Project (19)
               :                          :  :  +- Filter (18)
               :                          :  :     +- Scan csv  (17)
               :                          :  +- Project (22)
               :                          :     +- Filter (21)
               :                          :        +- Scan csv  (20)
               :                          +- BroadcastExchange (27)
               :                             +- Project (26)
               :                                +- Filter (25)
               :                                   +- Scan csv  (24)
               +- TakeOrderedAndProject (59)
                  +- Project (58)
                     +- BroadcastHashJoin Inner BuildRight (57)
                        :- Project (39)
                        :  +- Filter (38)
                        :     +- Scan csv  (37)
                        +- BroadcastExchange (56)
                           +- HashAggregate (55)
                              +- Exchange (54)
                                 +- HashAggregate (53)
                                    +- Project (52)
                                       +- BroadcastHashJoin Inner BuildRight (51)
                                          :- Union (46)
                                          :  :- Project (42)
                                          :  :  +- Filter (41)
                                          :  :     +- Scan csv  (40)
                                          :  +- Project (45)
```

```
:     +- Filter (44)
:        +- Scan csv  (43)
+- BroadcastExchange (50)
   +- Project (49)
      +- Filter (48)
         +- Scan csv  (47)
```

Listing 9: Physical Plan of Query 3 with broadcast hint

```
== Physical Plan ==
AdaptiveSparkPlan (24)
+- Sort (23)
   +- Exchange (22)
      +- HashAggregate (21)
         +- Exchange (20)
            +- HashAggregate (19)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (18)
                              +- Project (17)
                                 +- BatchEvalPython (16)
                                    +- Project (15)
                                       +- BroadcastHashJoin Inner BuildRight (14)
                                          :- Union (9)
                                          :  :- Project (5)
                                          :  :  +- Filter (4)
                                          :  :     +- Scan csv  (3)
                                          :  +- Project (8)
                                          :     +- Filter (7)
                                          :        +- Scan csv  (6)
                                          +- BroadcastExchange (13)
                                             +- Project (12)
                                                +- Filter (11)
                                                   +- Scan csv  (10)
```

Listing 10: Physical Plan of Query 4 -liable- with broadcast hint

```
== Physical Plan ==
AdaptiveSparkPlan (30)
+- Sort (29)
   +- Exchange (28)
      +- HashAggregate (27)
         +- Exchange (26)
            +- HashAggregate (25)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (24)
                              +- Filter (23)
                                 +- Window (22)
                                    +- WindowGroupLimit (21)
                                       +- Sort (20)
                                          +- Exchange (19)
                                             +- WindowGroupLimit (18)
                                                +- Sort (17)
```

```
+- Project (16)
   +- BatchEvalPython (15)
      +- BroadcastNestedLoopJoin Inner BuildRight (14)
         :- Project (10)
         :  +- Union (9)
         :     :- Project (5)
         :     :  +- Filter (4)
         :     :     +- Scan csv  (3)
         :     +- Project (8)
         :        +- Filter (7)
         :           +- Scan csv  (6)
         +- BroadcastExchange (13)
            +- Project (12)
               +- Scan csv  (11)
```

Listing 11: Physical Plan of Query 4 -closest- with broadcast hint

Οπως διακρίνεται στα παραπάνω physical plans, σε όλα τα join εφαρμόζονται BroadcastExchange με BroadcastHashJoin Inner BuildRight (βλέπε π.χ. στο listing 10 τα στάδια 13 και 14, αντίστοιχα), εκτός από το join του Query 4 που αφορά στα πλησιέστερα τμήματα (closest). Αυτό είναι αναμενόμενο, αφού όλα τα υπόλοιπα joins πλην αυτού έχουν ως condition κάποια ισότητα, προϋπόθεση για το broadcast hash join, (π.χ. αυτό του listing 10 έχει ως condition να ταυτίζονται οι τιμές του πεδίου "PREC"), ενώ στο join-εξαίρεση δεν εφαρμόζεται κάποιο condition (πρόκειται ουσιαστικά για καρτεσιανό γινόμενο), οπότε εφαρμόζεται broadcast nested loop, όπως βλέπουμε και στο αντίστοιχο physical plan: BroadcastNestedLoopJoin Inner BuildRight (14)

(Sort) Merge Join

Το (sort) merge join, όπως προδίδει το όνομά του, βασίζεται στο sorting των δεδομένων των δύο dataframes που πρόκειται να γίνουν joined. Συγκεκριμένα, γίνεται repartition και sorting των dataframes σύμφωνα με την τιμή-κλειδί (αυτή που αφορά στο condition του join). Έτσι, όχι μόνο χρειάζεται να ελεγχθούν για το join όσα records από τα δύο dataframes βρίσκονται σε ίδιο executor, αλλά είναι και sorted, οπότε η διαδικασία του join (το merge) είναι αποδοτικότερο. Ωστόσο, η στρατηγική αυτή μπορεί να εφαρμοστεί μόνο όπου το condition είναι έλεγχος ισότητας και υπάρχει η δυνατότητα για sorting της τιμής-κλειδί.

Τα physical plans των εκτελέσεων των Query 3 και 4 (liable και closest subquery) με μόνο merge hint είναι αντιστοίχως τα εξής:

```
== Physical Plan ==
AdaptiveSparkPlan (88)
+- Project (87)
   +- Sort (86)
      +- Exchange (85)
         +- HashAggregate (84)
            +- Exchange (83)
               +- HashAggregate (82)
                  +- Project (81)
                     +- SortMergeJoin Inner (80)
                        :- Sort (18)
                        :  +- Exchange (17)
                        :     +- Project (16)
```

```
                        :        +- SortMergeJoin Inner (15)
                        :           :- Sort (9)
                        :           :  +- Exchange (8)
                        :           :     +- Union (7)
                        :           :        :- Project (3)
                        :           :        :  +- Filter (2)
                        :           :        :     +- Scan csv  (1)
                        :           :        +- Project (6)
                        :           :           +- Filter (5)
                        :           :              +- Scan csv  (4)
                        :           +- Sort (14)
                        :              +- Exchange (13)
                        :                 +- Project (12)
                        :                    +- Filter (11)
                        :                       +- Scan csv  (10)
                        +- Sort (79)
                           +- Exchange (78)
                              +- Union (77)
                                 :- TakeOrderedAndProject (47)
                                 :  +- Project (46)
                                 :     +- SortMergeJoin Inner (45)
                                 :        :- Sort (23)
                                 :        :  +- Exchange (22)
                                 :        :     +- Project (21)
                                 :        :        +- Filter (20)
                                 :        :           +- Scan csv  (19)
                                 :        +- Sort (44)
                                 :           +- Exchange (43)
                                 :              +- HashAggregate (42)
                                 :                 +- Exchange (41)
                                 :                    +- HashAggregate (40)
                                 :                       +- Project (39)
                                 :                          +- SortMergeJoin Inner (38)
                                 :                             :- Sort (32)
                                 :                             :  +- Exchange (31)
                                 :                             :     +- Union (30)
                                 :                             :        :- Project (26)
                                 :                             :        :  +- Filter (25)
                                 :                             :        :
+- Scan csv  (24)
                                 :                             :        +- Project (29)
                                 :                             :           +- Filter (28)
                                 :                             :
+- Scan csv  (27)
                                 :                             +- Sort (37)
                                 :                                +- Exchange (36)
                                 :                                   +- Project (35)
                                 :                                      +- Filter (34)
                                 :                                         +- Scan csv
(33)
                                 +- TakeOrderedAndProject (76)
                                    +- Project (75)
                                       +- SortMergeJoin Inner (74)
                                          :- Sort (52)
                                          :  +- Exchange (51)
                                          :     +- Project (50)
                                          :        +- Filter (49)
                                          :           +- Scan csv  (48)
```

```
                                        +- Sort (73)
                                           +- Exchange (72)
                                              +- HashAggregate (71)
                                                 +- Exchange (70)
                                                    +- HashAggregate (69)
                                                       +- Project (68)
                                                          +- SortMergeJoin Inner (67)
                                                             :- Sort (61)
                                                             :  +- Exchange (60)
                                                             :     +- Union (59)
                                                             :        :- Project (55)
                                                             :        :  +- Filter (54)
                                                             :        :
+- Scan csv  (53)
                                                             :        +- Project (58)
                                                             :           +- Filter (57)
                                                             :
+- Scan csv  (56)
                                                             +- Sort (66)
                                                                +- Exchange (65)
                                                                   +- Project (64)
                                                                      +- Filter (63)
                                                                         +- Scan csv
(62)
```

Listing 12: Physical Plan of Query 3 with merge hint

```
== Physical Plan ==
AdaptiveSparkPlan (27)
+- Sort (26)
   +- Exchange (25)
      +- HashAggregate (24)
         +- Exchange (23)
            +- HashAggregate (22)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (21)
                              +- Project (20)
                                 +- BatchEvalPython (19)
                                    +- Project (18)
                                       +- SortMergeJoin Inner (17)
                                          :- Sort (11)
                                          :  +- Exchange (10)
                                          :     +- Union (9)
                                          :        :- Project (5)
                                          :        :  +- Filter (4)
                                          :        :     +- Scan csv  (3)
                                          :        +- Project (8)
                                          :           +- Filter (7)
                                          :              +- Scan csv  (6)
                                          +- Sort (16)
                                             +- Exchange (15)
                                                +- Project (14)
                                                   +- Filter (13)
                                                      +- Scan csv  (12)
```

Listing 13: Physical Plan of Query 4 -liable- with merge hint

```
== Physical Plan ==
AdaptiveSparkPlan (30)
+- Sort (29)
   +- Exchange (28)
      +- HashAggregate (27)
         +- Exchange (26)
            +- HashAggregate (25)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (24)
                              +- Filter (23)
                                 +- Window (22)
                                    +- WindowGroupLimit (21)
                                       +- Sort (20)
                                          +- Exchange (19)
                                             +- WindowGroupLimit (18)
                                                +- Sort (17)
                                                   +- Project (16)
                                                      +- BatchEvalPython (15)
                                                         +- BroadcastNestedLoopJoin Inner BuildRight (14)
                                                            :- Project (10)
                                                            :  +- Union (9)
                                                            :     :- Project (5)
                                                            :     :  +- Filter (4)
                                                            :     :     +- Scan csv  (3)
                                                            :     +- Project (8)
                                                            :        +- Filter (7)
                                                            :           +- Scan csv  (6)
                                                            +- BroadcastExchange (13)
                                                               +- Project (12)
                                                                  +- Scan csv  (11)
```

Listing 14: Physical Plan of Query 4 -closest- with merge hint

Η sort merge στρατηγική έχει την ίδια προϋπόθεση με την broadcast hash, το condition να αφορά σε ισότητα των τιμών-κλειδιά. Έτσι, τα joins του Query 3 και αυτό του subquery 4 που αφορά στα υπεύθυνα αστυνομικά τμήματα, τα οποία έχουν τέτοιου είδους condition, εφαρμόζουν την δηλωμένη στρατηγική (SortMergeJoin Inner). Αντιθέτως, όπως παρατηρήθηκε και στο αντίστοιχο physical plan του broadcast hint, στο join του query_4_closest (που όπως είδαμε υλοποιεί ουσιαστικά καρτεσιανό γινόμενο) δεν μπορεί να εφαρμοστεί αυτή η στρατηγική, οπότε ο optimizer (για λόγους που αναλύουμε στην συνέχεια) επιλέγει την BroadcastNestedLoopJoin Inner BuildRight (βλ. Listing 14).

Shuffle Hash Join

Το shuffle hash είναι μια στρατηγική όμοια με την broadcast hash "υπο-μέθοδο" της broadcast. Βασίζεται επίσης, για λόγους επίδοσης, στο hashing των δεδομένων των dataset, έχοντας γι' αυτόν τον λόγο τον ίδιο περιορισμό το condition να αφορά έλεγχο ισότητας. Ωστόσο, η μετάδοση των δεδομένων γίνεται με "shuffling", δηλαδή όλοι οι executors επικοινωνούν με όλους (all-to-all communication, μπορεί να οδηγήσει σε network congestion) για να μοιραστούν τα δεδομένα των δύο προς-join datasets.

Παρακάτω παρατίθενται τα physical plans των εκτελέσεων των Query 3 και 4 (liable και closest subquery) με shuffle hash hint, αντίστοιχα:

```
== Physical Plan ==
AdaptiveSparkPlan (76)
+- Project (75)
   +- Sort (74)
      +- Exchange (73)
         +- HashAggregate (72)
            +- Exchange (71)
               +- HashAggregate (70)
                  +- Project (69)
                     +- ShuffledHashJoin Inner BuildRight (68)
                        :- Exchange (15)
                        :  +- Project (14)
                        :     +- ShuffledHashJoin Inner BuildRight (13)
                        :        :- Exchange (8)
                        :        :  +- Union (7)
                        :        :     :- Project (3)
                        :        :     :  +- Filter (2)
                        :        :     :     +- Scan csv  (1)
                        :        :     +- Project (6)
                        :        :        +- Filter (5)
                        :        :           +- Scan csv  (4)
                        :        +- Exchange (12)
                        :           +- Project (11)
                        :              +- Filter (10)
                        :                 +- Scan csv  (9)
                        +- Exchange (67)
                           +- Union (66)
                              :- TakeOrderedAndProject (40)
                              :  +- Project (39)
                              :     +- ShuffledHashJoin Inner BuildRight (38)
                              :        :- Exchange (19)
                              :        :  +- Project (18)
                              :        :     +- Filter (17)
                              :        :        +- Scan csv  (16)
                              :        +- Exchange (37)
                              :           +- HashAggregate (36)
                              :              +- Exchange (35)
                              :                 +- HashAggregate (34)
                              :                    +- Project (33)
                              :                       +- ShuffledHashJoin Inner BuildRight (32)
                              :                          :- Exchange (27)
                              :                          :  +- Union (26)
                              :                          :     :- Project (22)
                              :                          :     :  +- Filter (21)
                              :                          :     :     +- Scan csv  (20)
                              :                          :     +- Project (25)
                              :                          :        +- Filter (24)
                              :                          :           +- Scan csv  (23)
                              :                          +- Exchange (31)
                              :                             +- Project (30)
                              :                                +- Filter (29)
                              :                                   +- Scan csv  (28)
                              +- TakeOrderedAndProject (65)
                                 +- Project (64)
                                    +- ShuffledHashJoin Inner BuildRight (63)
```

```
:- Exchange (44)
:  +- Project (43)
:     +- Filter (42)
:        +- Scan csv  (41)
+- Exchange (62)
   +- HashAggregate (61)
      +- Exchange (60)
         +- HashAggregate (59)
            +- Project (58)
               +- ShuffledHashJoin Inner BuildRight (57)
                  :- Exchange (52)
                  :  +- Union (51)
                  :     :- Project (47)
                  :     :  +- Filter (46)
                  :     :     +- Scan csv  (45)
                  :     +- Project (50)
                  :        +- Filter (49)
                  :           +- Scan csv  (48)
                  +- Exchange (56)
                     +- Project (55)
                        +- Filter (54)
                           +- Scan csv  (53)
```

Listing 15: Physical Plan of Query 3 with shuffle_hash hint

```
== Physical Plan ==
AdaptiveSparkPlan (25)
+- Sort (24)
   +- Exchange (23)
      +- HashAggregate (22)
         +- Exchange (21)
            +- HashAggregate (20)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (19)
                              +- Project (18)
                                 +- BatchEvalPython (17)
                                    +- Project (16)
                                       +- ShuffledHashJoin Inner BuildRight (15)
                                          :- Exchange (10)
                                          :  +- Union (9)
                                          :     :- Project (5)
                                          :     :  +- Filter (4)
                                          :     :     +- Scan csv  (3)
                                          :     +- Project (8)
                                          :        +- Filter (7)
                                          :           +- Scan csv  (6)
                                          +- Exchange (14)
                                             +- Project (13)
                                                +- Filter (12)
                                                   +- Scan csv  (11)
```

Listing 16: Physical Plan of Query 4 -liable- with shuffle_hash hint

```
== Physical Plan ==
AdaptiveSparkPlan (30)
+- Sort (29)
   +- Exchange (28)
      +- HashAggregate (27)
         +- Exchange (26)
            +- HashAggregate (25)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (24)
                              +- Filter (23)
                                 +- Window (22)
                                    +- WindowGroupLimit (21)
                                       +- Sort (20)
                                          +- Exchange (19)
                                             +- WindowGroupLimit (18)
                                                +- Sort (17)
                                                   +- Project (16)
                                                      +- BatchEvalPython (15)
                                                         +- BroadcastNestedLoopJoin Inner BuildRight (14)
                                                            :- Project (10)
                                                            :  +- Union (9)
                                                            :     :- Project (5)
                                                            :     :  +- Filter (4)
                                                            :     :     +- Scan csv  (3)
                                                            :     +- Project (8)
                                                            :        +- Filter (7)
                                                            :           +- Scan csv  (6)
                                                            +- BroadcastExchange (13)
                                                               +- Project (12)
                                                                  +- Scan csv  (11)
```

Listing 17: Physical Plan of Query 4 -closest- with shuffle_hash hint

Και η shuffle hash στρατηγική για join (που άλλωστε διαφέρει από την broadcast hash μόνο στην μέθοδο διαμοιρασμού της πληροφορίας) έχει τον περιορισμό η συνθήκη του join να αφορά σε ισότητα. Οπότε είναι αναμενόμενο, και επιβεβαιώνεται από τα παραπάνω physical plans, τα joins εκτός του τελευταίου (του μοναδικού του query_4_closest) να υλοποιούν την shuffle hash στατηγική, ενώ το τελευταίο να υλοποιεί, για τον ίδιο λόγο με τις προηγούμενες στρατηγικές, broadcast nested loop.

Shuffle and Replicate Nested Loop Join

Γνωστή και ως Cross Project Join (CPJ), η μέθοδος αυτή μοιάζει με την shuffle hash, διαφέροντας μόνο στην μη χρήση hash. Αντί της δημιουργίας hash tables από τα datasets, μεταδίδονται αντίγραφα των datasets (με all-to-all επικοινωνία των executors).

Τα physical plans των εκτελέσεων των Query 3 και 4 (liable και closest subquery) με μόνο merge hint είναι αντιστοίχως τα εξής:

```
== Physical Plan ==
AdaptiveSparkPlan (64)
+- Project (63)
   +- Sort (62)
      +- Exchange (61)
```

```
+- HashAggregate (60)
   +- Exchange (59)
      +- HashAggregate (58)
         +- Project (57)
            +- CartesianProduct Inner (56)
               :- Project (12)
               :  +- CartesianProduct Inner (11)
               :     :- Union (7)
               :     :  :- Project (3)
               :     :  :  +- Filter (2)
               :     :  :     +- Scan csv  (1)
               :     :  +- Project (6)
               :     :     +- Filter (5)
               :     :        +- Scan csv  (4)
               :     +- Project (10)
               :        +- Filter (9)
               :           +- Scan csv  (8)
               +- Union (55)
                  :- TakeOrderedAndProject (33)
                  :  +- Project (32)
                  :     +- CartesianProduct Inner (31)
                  :        :- Project (15)
                  :        :  +- Filter (14)
                  :        :     +- Scan csv  (13)
                  :        +- HashAggregate (30)
                  :           +- Exchange (29)
                  :              +- HashAggregate (28)
                  :                 +- Project (27)
                  :                    +- CartesianProduct Inner (26)
                  :                       :- Union (22)
                  :                       :  :- Project (18)
                  :                       :  :  +- Filter (17)
                  :                       :  :     +- Scan csv  (16)
                  :                       :  +- Project (21)
                  :                       :     +- Filter (20)
                  :                       :        +- Scan csv  (19)
                  :                       +- Project (25)
                  :                          +- Filter (24)
                  :                             +- Scan csv  (23)
                  +- TakeOrderedAndProject (54)
                     +- Project (53)
                        +- CartesianProduct Inner (52)
                           :- Project (36)
                           :  +- Filter (35)
                           :     +- Scan csv  (34)
                           +- HashAggregate (51)
                              +- Exchange (50)
                                 +- HashAggregate (49)
                                    +- Project (48)
                                       +- CartesianProduct Inner (47)
                                          :- Union (43)
                                          :  :- Project (39)
                                          :  :  +- Filter (38)
                                          :  :     +- Scan csv  (37)
                                          :  +- Project (42)
                                          :     +- Filter (41)
                                          :        +- Scan csv  (40)
                                          +- Project (46)
```

```
                                                  +- Filter (45)
                                                     +- Scan csv  (44)
```

Listing 18: Physical Plan of Query 3 with shuffle_replicate_nl hint

```
== Physical Plan ==
AdaptiveSparkPlan (24)
+- Sort (23)
   +- Exchange (22)
      +- HashAggregate (21)
         +- Exchange (20)
            +- HashAggregate (19)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (18)
                              +- Project (17)
                                 +- BatchEvalPython (16)
                                    +- Project (15)
                                       +- BroadcastHashJoin Inner BuildRight (14)
                                          :- Union (9)
                                          :  :- Project (5)
                                          :  :  +- Filter (4)
                                          :  :     +- Scan csv  (3)
                                          :  +- Project (8)
                                          :     +- Filter (7)
                                          :        +- Scan csv  (6)
                                          +- BroadcastExchange (13)
                                             +- Project (12)
                                                +- Filter (11)
                                                   +- Scan csv  (10)
```

Listing 19: Physical Plan of Query 4 -liable- with shuffle_replicate_nl hint

```
== Physical Plan ==
AdaptiveSparkPlan (29)
+- Sort (28)
   +- Exchange (27)
      +- HashAggregate (26)
         +- Exchange (25)
            +- HashAggregate (24)
               +- InMemoryTableScan (1)
                     +- InMemoryRelation (2)
                           +- AdaptiveSparkPlan (23)
                              +- Filter (22)
                                 +- Window (21)
                                    +- WindowGroupLimit (20)
                                       +- Sort (19)
                                          +- Exchange (18)
                                             +- WindowGroupLimit (17)
                                                +- Sort (16)
                                                   +- Project (15)
                                                      +- BatchEvalPython (14)
                                                         +- CartesianProduct Inner (13)
                                                            :- Project (10)
```

```
:  +- Union (9)
:     :- Project (5)
:     :  +- Filter (4)
:     :     +- Scan csv  (3)
:     +- Project (8)
:        +- Filter (7)
:           +- Scan csv  (6)
+- Project (12)
   +- Scan csv  (11)
```

Listing 20: Physical Plan of Query 4 -closest- with shuffle_replicate_nl hint

Η Shuffle and Replicate Nested Loop στρατηγική, αποφεύγοντας την χρήση hashes, δεν έχει κάποιο περιορισμό ως προς το condition του join. Έτσι, περιμένουμε σε όλα τα joins των εν λόγω queries να χρησιμοποιείται όντως αυτή, κάτι που συμβαίνει όπως διακρίνεται και στα παραπάνω physical plans (CartesianProduct Inner).

Σύγκριση στρατηγικών

Όπως είδαμε παραπάνω, είναι σαφές ότι κάθε στρατηγική είναι υλοποιήσιμη και αποδοτικότερη από τις υπόλοιπες σε συγκεκριμένες εφαρμογές. Οι broadcast hash, (sort) merge και shuffle hash απαιτούν το condition του join να αφορά στην ισότητα των τιμών-κλειδιά, με την πρώτη, ενώ σαφώς αποδοτικότερη - αφού το broadcasting είναι καλύτερο από το repartition & sorting της merge και από το shuffling της shuffle hash - να απαιτεί το ένα dataframe να είναι αρκετά μικρό (τόσο ώστε να χωράει στην μνήμη των executors). Μεταξύ δε των δύο τελευταίων, η merge είναι αποδοτικότερη, εφόσον μετά το partitioning & sorting των δεδομένων το join (που δεν είναι σημαντικά πιο αργό από το shuffling των δεδομένων) απαιτούνται λιγότεροι έλεγχοι του condition του join, ωστόσο περιορίζεται σε εκείνες τις "εφαρμογές" στις οποίες το πεδίο του conditon μπορεί να ταξινομηθεί. Και οι τρεις, ωστόσο, χάρη στο hashing, αποτελούν καλύτερες επιλογές από τις broadcast nested loop και shuffle and replicate nested loop, η χρήση των οποίων δύο, όμως, είναι μονόδρομος αν το join δεν αφορά σε ισότητα της τιμής-κλειδιού. Στην περίπτωση αυτή, η broadcast nested loop προτιμάται εάν το ένα εκ των δύο dataframes είναι αρκετά μικρό (αφού μόνο τότε είναι υλοποιήσιμη).

Το τελευταίο, μπορούμε να το διαπιστώσουμε και από τα παραπάνω physical plans, όπου όποτε κάνουμε hint μια στρατηγική, εφαρμόσιμη μόνο σε condition ισότητας, σε join με διαφορετικό condition (π.χ. sort merge ή shuffle hash στο query_4_closest) προτιμάται, από τον optimizer, να υλοποιείται η Broadcast Nested Loop (η οποία έχοντας "μικρό" dataframe, το LAPD_Police_Stations_df, είναι εφαρμόσιμη).

Τέλος, κάποιες από τις ανωτέρω παρατηρήσεις επιβεβαιώνονται και από τις επιλογές του optimizer, όπως αυτές διακρίνονται στα plans των queries, τόσο σε μορφή κειμένου όσο και γραφικά στο Spark History UI. Παρακάτω, παρατίθενται αυτά στην δεύτερη εκ των δύο μορφή:

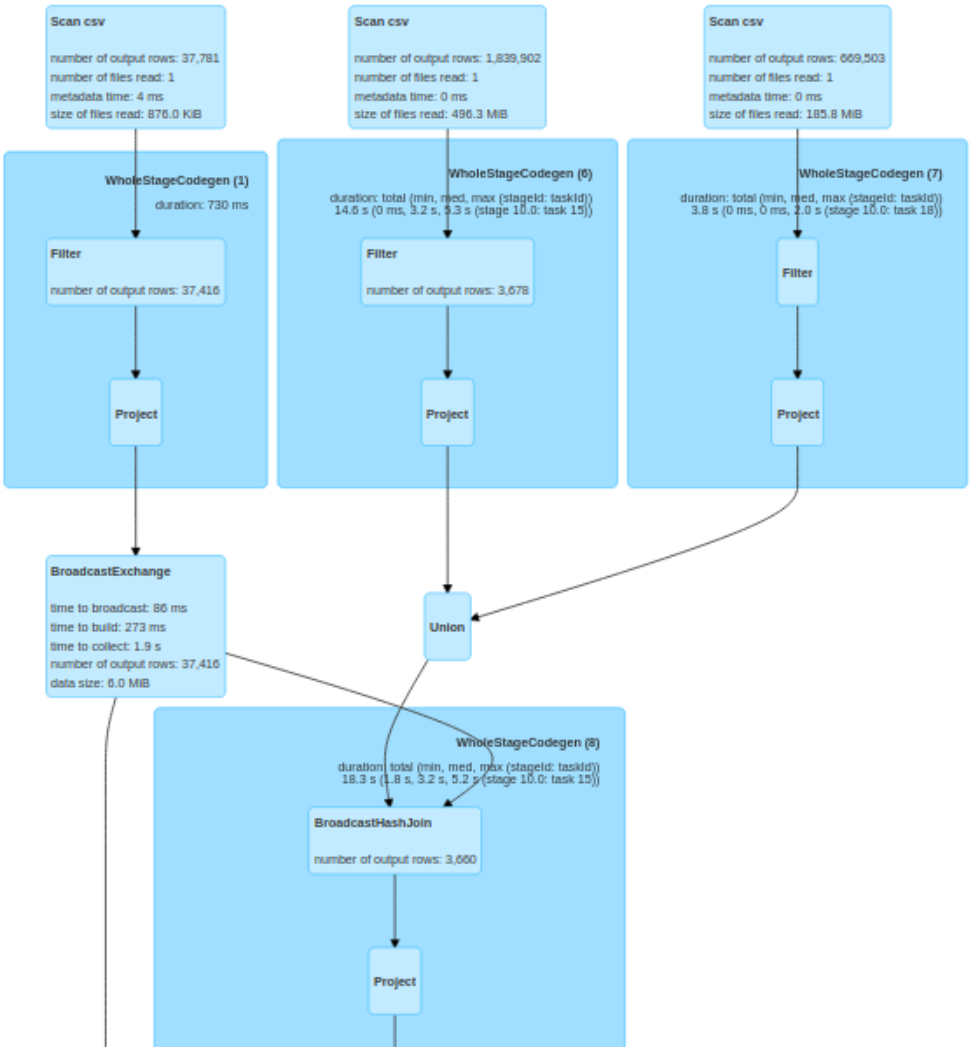

Figure 4: Query 3, 1st join

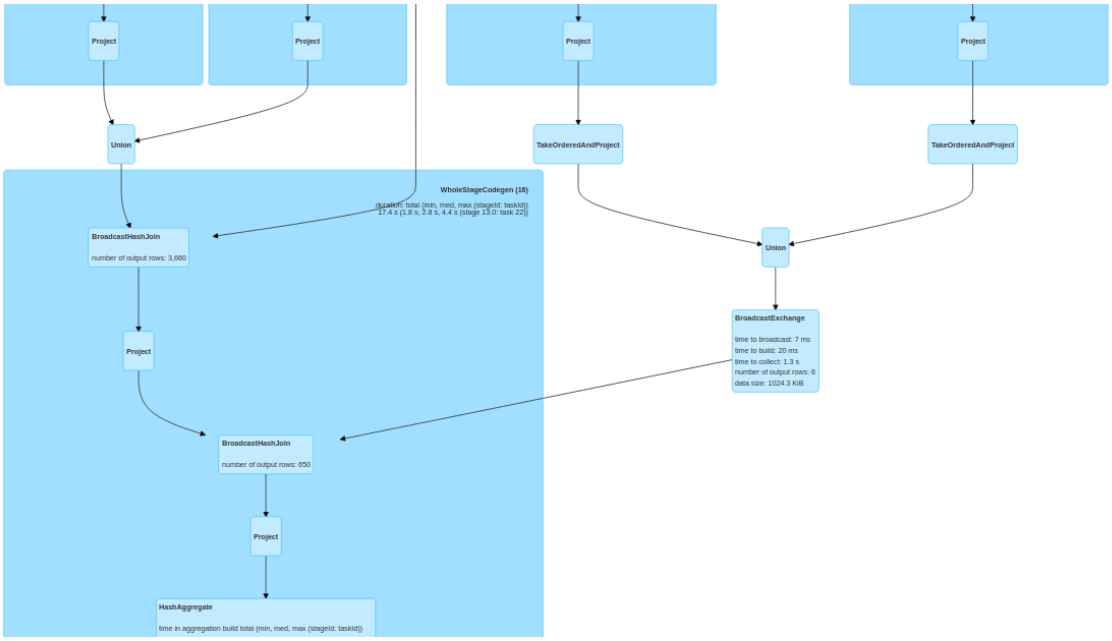

Figure 5: Query 3, 2nd and 3rd join

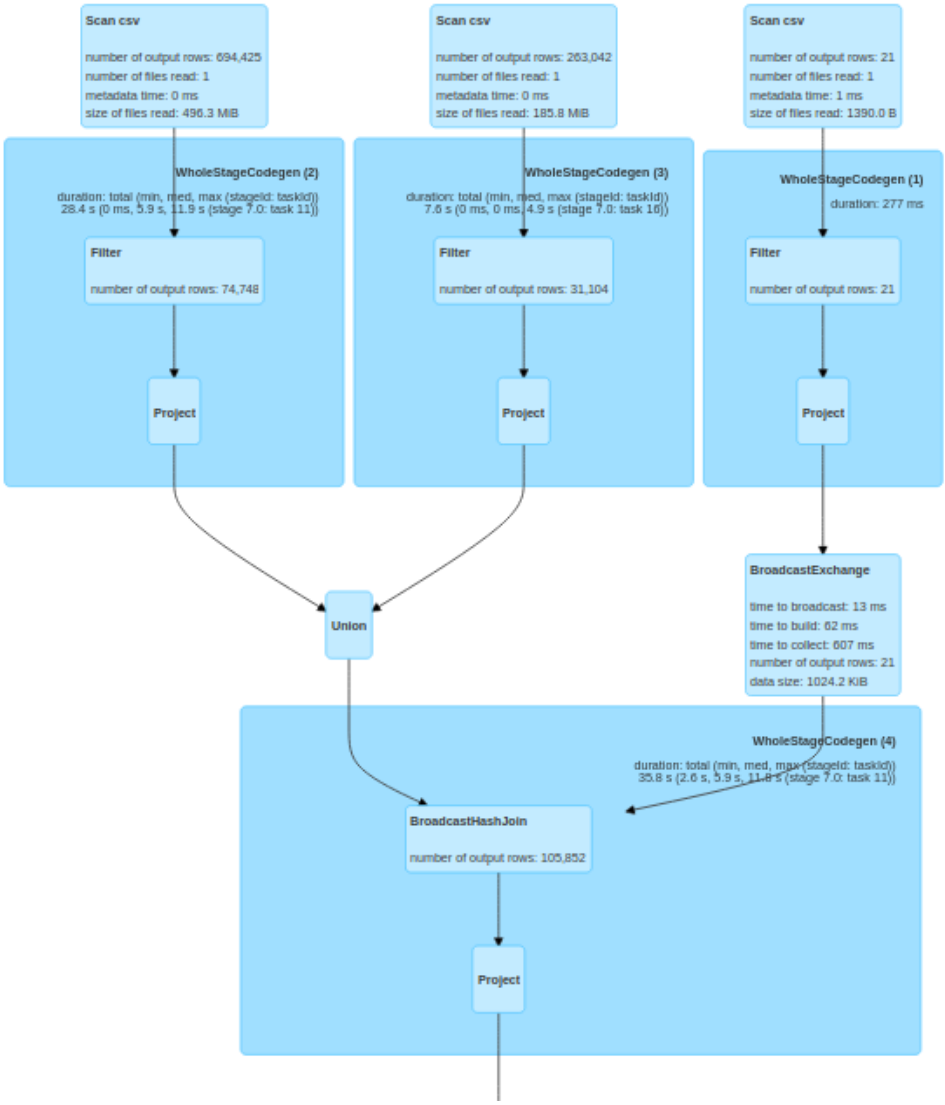

Figure 6: Query 4 -liable- join

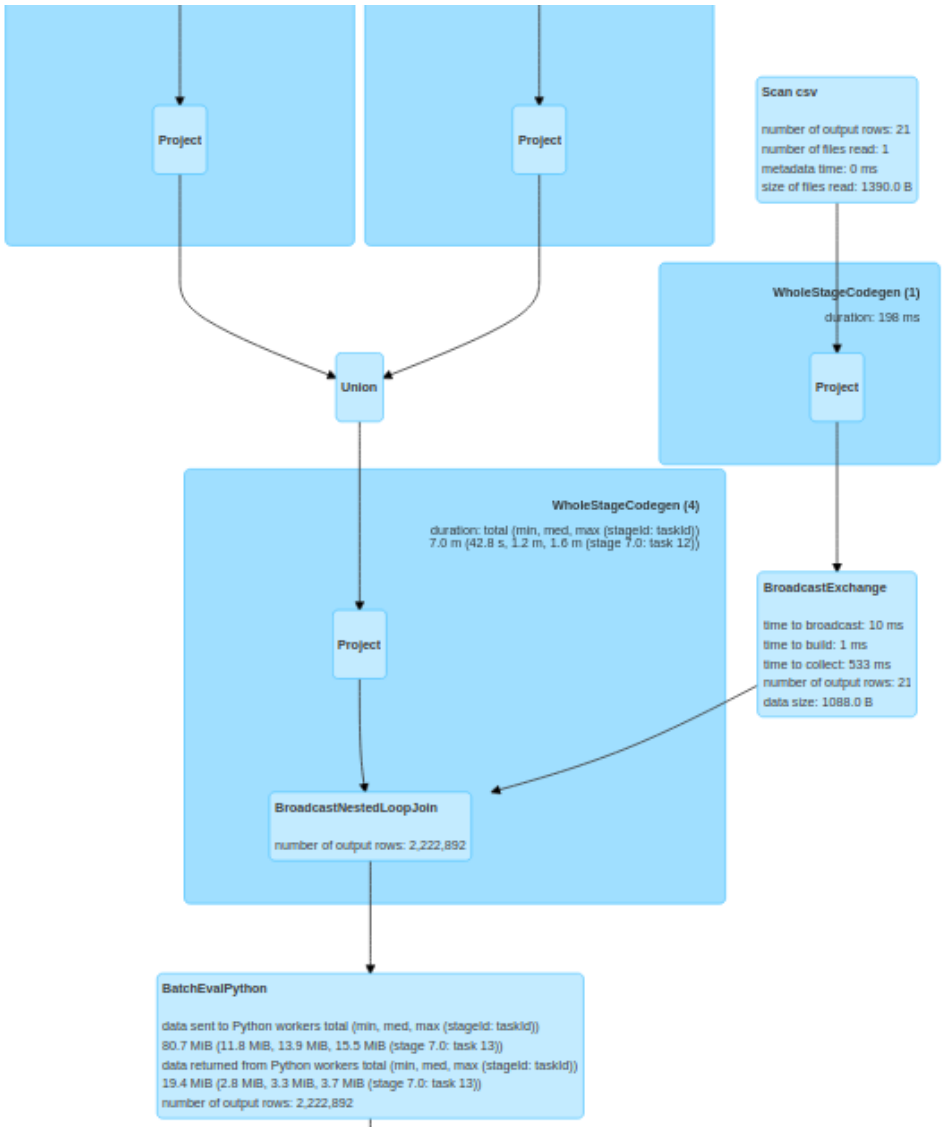

Figure 7: Query 4 -closest- join